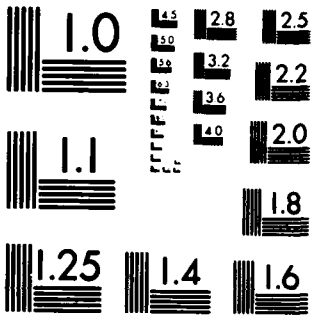
AD-A126 228 PRISM: A GENERAL PURPOSE PROGRAMMING SYSTEM(U) AIR FORCE HUMAN RESOURCES LAB BROOKS AFB TX C R ROGERS ET AL. MAR 83 AFHRL-TP-82-44 1/1

UNCLASSIFIED F/G 9/2 NL

END

DATE FILMED

DTIC



AFHRL-TP-82-44

(10)

AIR FORCE HUMAN RESOURCES LABORATORY

AD A126228

PRISM:
A GENERAL PURPOSE PROGRAMMING SYSTEM

By

Charles R. Rogers
Steven A. O'Hara

TECHNICAL SERVICES DIVISION
Brooks Air Force Base, Texas 78235

March 1983

Final Technical Paper

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

WENDELL L. ANDERSON, Lt Col, USAF
Chief, Technical Services Division

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER AFHRL-TP-82-44	2. GOVT ACCESSION NO. AD-A126228
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)* PRISM: A GENERAL PURPOSE PROGRAMMING SYSTEM	5. TYPE OF REPORT & PERIOD COVERED Final
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles R. Rogers Steven A. O'Hara	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Technical Services Division Air Force Human Resources Laboratory Brooks Air Force Base, Texas 78235	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62703F 63230423
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235	12. REPORT DATE March 1983
	13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS *(of this report)* Unclassified
	15.a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of this abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

artificial intelligence	facilities management
binding time	general purpose programs
computing algorithms	interactive debugging
data structures	interpreter
dynamic program structures	operating system interface

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This paper describes the development, uses, and features of the general purpose programming system PRISM, which is the foundation for future program development by the Computer Programming Branch and is available to all personnel within the Air Force Human Resources Laboratory (AFHRL). PRISM was designed to meet the need for an efficient and reliable programming tool that could be used like a high-order programming language but still provide the operating system interface and hardware controls of assembly language. It has special features that make

DD Form 1473 EDITION OF 1 NOV 65 IS OBSOLETE
1 Jan 73

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Date Entered)*

Item 19 (*Continued*)

PRISM
programming language
quality control
runstream generator
standard products
subroutine libraries
UNIVAC 1100
universal file handler
utility programs

Item 20 (*Continued*)

it an especially powerful tool for new software development. These features were derived from an extensive analysis of coding sequences in existing library programs, interactions between library programs, and the identification of common programming procedures. PRISM was specifically designed for the development of general purpose programs by the Technical Services Division of AFHRL within the Computer Programming Branch; however, it is also an effective and efficient tool for applications programmers.

AFHRL Technical Paper 82-44 March 1983

PRISM:
A GENERAL PURPOSE PROGRAMMING SYSTEM

Charles R. Rogers
Steven A. O'Hara

Software Development Section
Computer Programming Branch
Technical Services Division

Reviewed and submitted for publication by

Jimmy D. Souter
Chief, Computer Programming Branch

TABLE OF CONTENTS

		Page
I.	Introduction	3
	Need for General Purpose Programs	3
	Evolution of PRISM	3
	Quality Control Features	4
	Flexibility	5
	Maintenance	5
II.	Potential Uses of PRISM	7
	For General Purpose Programs	7
	For Special Purpose Programs	7
	As an Interactive Runstream Generator	8
III.	Special Features of PRISM	9
	Data File Access	9
	Operating System Interface	10
	User Definable Program Structures	10
	Error Handling and Interactive Debugging	11
IV.	Applications	13
	Current Uses	13
	Future Uses	13

PRISM: A GENERAL PURPOSE PROGRAMMING SYSTEM

I. INTRODUCTION

This paper describes the development, uses and features of PRISM, a general purpose programming system. As the newest and most versatile general purpose program, PRISM is the foundation for future program development by the Computer Programming Branch and is available to all personnel within the Air Force Human Resources Laboratory (AFHRL).

Need for General Purpose Programs

Customer Base. The Computer Programming Branch of the Technical Services Division provides data processing support to a wide range of customers, such as AFHRL research scientists, Air Force personnel managers, and researchers from other major commands. This has led to the development of large general purpose program libraries that enable applications programmers to produce a diverse, but standardized, set of computer products.

Increasing Workload. Since customers gain greater productivity through the use of standard products, there is a natural increase in the number of product requests. The continued use of standard products is reinforced in two ways. First, the customer usually requests the product by name. Second, it is faster and more reliable to produce standard products via general purpose programs than to write special purpose programs.

Software Requirements: At AFHRL, the programmer/analyst staff has remained relatively static for several years while the number of customers and volume of product requests have continued to increase. Some of the additional time required to process the increasing number of computer runs has been reduced by the installation of faster, more reliable hardware. The remainder of the time must be made up by more efficient software, both in terms of computational speed and reduction of the number of work steps necessary to produce a standard product.

Evolution of PRISM

Special Purpose Programs. A special purpose program is a solution to a specific problem. When another request is received for a similar special purpose product using different source data, a programmer manually modifies the program to get the new product. If requested often enough, the product then becomes a "standard product" that can be requested by a generic product name.

Standard Products. While the content of standard products may vary, the general format of the products is the same. This format would have been developed jointly by the customer and the programmer. Interpretation of the product by the original customer requires no additional training, and training for other customers can be standardized.

The Generalization Process. When the original program author leaves AFHRL, each standardized special purpose program is turned over to the Software Development Section of the Computer Programming Branch to determine if it should become generalized and placed in the general purpose program library. A detailed analysis of the special purpose program is made to separate the constant data and logic. Control card commands are then designed to enable the user to input the variable information with the least amount of effort. A new program is then developed that will "learn" the variable part of the task from the control cards and then use that information to perform the task plus all the necessary quality control operations. For example, while a special purpose program must know the exact file and record formats at compile time, a general purpose program learns this information at run time.

General Purpose Programs. The use of general purpose programs is the most effective means of increasing programmer productivity. Significantly less time is required to code general purpose control cards than to code program language statements. General purpose programs do not require the iterative compile and test steps necessary to the development of special purpose programs. Decision making is elevated to global system problems with little time spent on the mechanics of input/output (I/O) and data manipulation. Quality control is streamlined for individual work steps.

PRISM Design. The design of PRISM began with the analysis of existing general purpose programs to isolate the standardized operations common to more than one program. Some of these common operations were placed into a subroutine library while others were incorporated into PRISM as instructions, functions, or system variables. The result is consolidation and standardization of virtually all the computing algorithms necessary for general purpose program development into a single modular programming system.

Quality Control Features

Computing Algorithms. Rigorous quality control of computing algorithms is performed during the development of a general purpose program. This reduces the time necessary to audit individual work steps when general purpose programs are used in production runs. Increased quality control productivity is achieved by elevating the decision making to the overall system level. For example, an auditor can concentrate on whether the correct statistic was used rather than on the computational accuracy of the statistic.

Logic Tracing. Quality control productivity is further increased by automating tedious logic tracing procedures needed to prove the accuracy of the individual work steps. All general purpose programs include such things as I/O counts, true/false counts at major decision points, and descriptive diagnostics. Since quality control personnel are involved in the program development process, they are in a position to define specific audit aids.

Flexibility

Systems Design. The data processing for research studies can be performed by linking several general purpose programs into a chain. As a general rule, the more complex studies require more program links. Since each general purpose program evolved from a special purpose program, there are times when extra links must be inserted in the chains to transform intermediate data files from one format to another. PRISM is the first of the general purpose programs to have true file independence; that is, the remaining obstacles to total flexibility in arranging the program links are removed.

Binding Time. Binding time refers to the instant when a symbolic expression in a computer program is reduced to a form directly interpretable by the hardware. Program flexibility increases as the binding time of program information is shifted from compile to run time. Because PRISM can bind new control statements while it is executing, program logic can actually be changed while a program is running. This gives the programmer the ability to take advantage of artificial intelligence (AI) techniques. PRISM is so flexible that it can be, and is, used to create other general purpose programs.

Maintenance

Simplicity. As a general rule, the number of statements in a symbolic PRISM program is less than half those in a comparable COBOL or FORTRAN program and less than a tenth of a comparable assembly language program. A very large part of all general purpose programs is the overhead necessary to interpret control statements at run time. By providing powerful, built-in, free format scanning functions, PRISM simplifies this activity, reduces the overhead coding and stabilizes control statement syntax. Maintenance programming is thereby simplified and standardized across all general purpose programs.

Standardization. Coupled with standard programming conventions, PRISM programs are significantly easier to maintain because the powerful built-in functions reduce the tendency for programmers to "re-invent the wheel." Experience with writing general purpose programs in PRISM and using standard techniques has proved that

programs written by one programmer can be modified by another person with little or no interaction between the two people. The obvious benefits include a reduction in costly maintenance time and the ability to apply a standard set of updates to a large number of general purpose programs when changes to the operating environment dictate.

II. POTENTIAL USES OF PRISM

Although PRISM was specifically designed for the development of general purpose programs by the Software Development Section, it is also an effective and efficient tool for applications programmers. Because of its flexibility and power, it can be thought of as a stand-alone general purpose program, a high-order language for writing both special purpose and general purpose programs, and an interactive runstream generator.

For General Purpose Programs

Programming Language. Virtually any algorithm that can be expressed in any programming language can be expressed with PRISM control statements. Besides incorporating the basic features from other high-order languages, PRISM contains many unique features, some of which are described later.

Utility Processors. PRISM has a special set of features which allow programmers to create utility programs that perform like compiled UNIVAC system processors. Applications personnel can execute a utility processor and provide all pertinent control information with a single processor call statement.

Examples. An increasing percentage of the programs written and maintained by the Software Development Section are coded in PRISM. These include the documentation retrieval system, a frequency distribution reporting program, a generalized sort program, and a utility file query program.

For Special Purpose Programs

One-Time Runs. There are occasions when a programmer needs to perform a unique task and knows that this specific task will not be performed again. Because PRISM is an interpreter, tasks of this type can be easily coded and immediately executed without the time-consuming cycles of compiling and testing.

Chained Runstreams. The ability of PRISM to access data in any file format can be used to create links which perform special data transformations for the other programs in a chained runstream. PRISM can also be used to create temporary communications files for passing information between the other programs in the chain.

Examples. PRISM's file handling capability can be used to create test data before the arrival of actual data. This enables programmers to write and test their programs without delay. PRISM can efficiently extract and summarize selected portions of a print file, thereby reducing both paper costs and time.

As an Interactive Runstream Generator

Pre-Compiler. The versatile string manipulation capabilities of PRISM make it an excellent tool for coding pre-compiler languages whereby the source statements are transformed into valid input for some other language processor such as COBOL. This provides a very powerful macro process with comprehensive error checking capabilities.

Runstream Prototypes. When a task is to be executed several times, with similar but not identical input, a PRISM program can be used to insert the variable information into a predefined prototype runstream. This process includes the prompting for all necessary information and quality control of the information received. The generated runstream can be started as an asynchronous batch activity or executed immediately.

Examples. The COBOL pre-compiler (VGN) is being rewritten in PRISM to take advantage of the upgraded COBOL compiler features, increase efficiency, and provide additional flexibility. The summary statistics package (STATPK) is both a FORTRAN pre-compiler and a runstream generator. The general purpose report writer (RPT) is a PRISM program that learns the report writing task and replaces itself with a generated PRISM program which performs the learned task.

III. SPECIAL FEATURES OF PRISM

PRISM has several outstanding features that make it an especially powerful tool for the development of general purpose programs. These features were developed after extensive analysis of coding sequences in existing library programs, interactions between sequences in existing library programs, interactions between library programs, and the identification of common programming procedures. In a stepwise refinement process, new programs written in PRISM were examined for coding patterns that could be reduced to single instructions or functions by making enhancements to PRISM.

Data File Access

File Handling. The incompatibility of data file formats is a common attribute of UNIVAC 1100 compiler languages. UNIVAC system data format (SDF) files cannot be readily accessed by COBOL or FORTRAN programs. A COBOL program cannot access a FORTRAN file, and vice versa. While this has little effect on one-time special purpose programs, general purpose programs must be able to access data from any reasonable source. PRISM solves this problem by its ability to read virtually any file format and to write more than one standard format.

Input/Output Control. Under certain circumstances, it is desirable for a program to do more than simply "read the next record" or "write a record." PRISM maintains communications packets as string variables and allows the user program to place information into or receive information from these packets. This gives the programmer complete control over I/O functions such as the processing of file labels and simulated random access of sequential input files.

Run-Time File Definition. While language processors allow multiple inputs and outputs, the number of files and file characteristics are fixed at compile time. PRISM carries the file definition concept one step further by associating a channel number with each I/O file. This channel number can be a variable, and any number of channels may be open at any time. The I/O commands and communications packets are linked by this channel number. All packets and buffers are created upon first reference, thus keeping the central memory space to a minimum.

Non-Standard Files. In addition to the normal operations of read, write, and print, PRISM provides several specialized operations for accessing arbitrary units of mass storage files as logical units. The table of contents for a program file is accessible as though it were also a file.

Facilities Management. PRISM dynamically assigns files at run time and returns them to their original state when closed. The user program is in complete control of file errors. If a file is not explicitly closed by the user program, PRISM automatically closes it at the termination of execution.

Operating System Interface

Run Information. With the exception of the current date and time, UNIVAC language processors do not provide immediate access to other system information, such as the user's run identification, site identification, or breakpoint status. PRISM provides two forms of predefined variables to accommodate those programs that need this information. Some variables, such as user's account number or project identification, are static. Other variables, such as the current time of day or the user's name, are computed when referenced.

System Information. Historically, one of the main reasons for writing assembler subroutines was to access information available only through operating system interface routines called Executive Requests (ERs). PRISM has special instructions and intrinsic functions to provide direct access to the Program Control Table (PCT), Master File Directory (MFD), and other system information.

System Processor Call. PRISM provides a mechanism whereby a program may be converted to a standard system processor. The program can then be executed like any other processor without the user's having to know that PRISM is in control. The entire processor call image (INFOR) is available to the program as a string variable. There are special instructions and functions available to facilitate the interpretation of the INFOR image.

Runstream Control. For use as a front end processor, PRISM has a special set of instructions to enable a program to queue other programs for execution following termination of the PRISM program.

User Definable Program Structures

Dynamic Instruction Generation. PRISM is unique with respect to artificial intelligence (AI) programming techniques. There are three different means for creating and executing PRISM instructions at run time.

1. The EX command interprets and executes a string expression as though it were a PRISM instruction.

2. The INS command inserts string expressions as PRISM instructions into the program as it is being executed. There is a corresponding command to delete PRISM instructions.

3. The LINK command replaces the entire program being executed by another PRISM program, which also could have been created at run time.

Dynamic Structures. The nature of a compiler is such that program structures are defined as much as possible at compile time. Neither COBOL nor FORTRAN provides any mechanism for dynamic data structures. Because PRISM is an interpreter, it has no such restrictions. With a few simple commands, and the use of pointer variables, virtually any data structure can be created and easily accessed.

Predefined Structures. In addition to the general case, PRISM has several built-in data structures. These stacks, queues, and binary trees represent those structures that are most commonly used and that serve as efficient and reliable tools for general purpose program development.

1. Stacks used on internal subroutine calls provide recursive programming capabilities.

2. Queuing operations allow both first-in first-out and last-in first-out queues to be used.

3. Binary tree functions provide optional capabilities for such things as in-core sorts and frequency distributions.

Symbolic Substring Definition. Any PRISM string variable can be treated as a record containing binary values, ASCII characters, and/or FIELDATA characters. The subfield definition (FLD) allows treating any portion of a record as a separate variable. The ability to change the value of a subfield definition at run time provides another degree of flexibility.

Error Handling and Interactive Debugging

Descriptive Diagnostics. Because a program rarely runs perfectly the first time, a well designed processor must provide meaningful diagnostic and debugging aids. PRISM has two sets of diagnostics, one set for before execution starts and the other for run-time errors. The user of PRISM is never left stranded by an aborted program without a descriptive error message.

Dump Mode. The dump mode is a special option that assists in debugging a program by decoding the entire program back to symbolic form before or during execution. It is used to verify that PRISM has correctly interpreted the symbolic input.

Trace Mode. A program controllable trace mode in PRISM provides run-time tracing of both data and instructions. In this mode, a line is printed before each instruction is executed, showing diagnostic information about the flow of execution along with any user-defined message.

Execute Mode. An interactive debugging execute mode gives complete program control to the programmer with options available for controlling program execution, including single-stepping. At each step, the programmer can enter any PRISM instruction for immediate execution.

IV. APPLICATIONS

PRISM was specifically designed to meet the need for an efficient and reliable programming tool that could be used like a high-order programming language but still provide the operating system interface and hardware controls of assembly language.

Current Uses

Replace Assembly Language. Utility programs that previously were required to be written in assembly language are now being replaced by PRISM programs that have the same level of control over the machine.

Standardization. All new general purpose and utility programs are written in PRISM. Each program that must be upgraded due to new requirements or system changes is being converted to PRISM. This standardization process also includes option letters and specification fields on processor call statements, prompting messages, and diagnostic messages.

Future Uses

New Algorithms. Because PRISM was designed and written as a symbolic interpreter consisting of independent modules for each instruction and function, creation and insertion of new computing algorithms can be rapidly and efficiently implemented. That is, the system was designed to be changed so that PRISM can remain current with state-of-the-art programming practices, operating system changes, and new equipment.

Conversion. As an interpreter, PRISM can be written in any programming language in the event of a complete computer system replacement. The general purpose and utility programs written in the PRISM language would not have to be rewritten beyond those changes necessary to conform to new requirements such as file naming conventions. This capability protects AFHRL's investment in software development and maintenance.